Программирование

Python

Кафедра ИВТ и ПМ

2017

План

Синтаксис

Синтаксис языка программирования — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом.

Синтаксис

Пример синтаксического правила.

Для вызова функции с параметром x следует указать имя функции (её идентификатор), затем в круглых скобках выражение - пераметр. Допустимо ставить пробелы между идентификаторами и скобками.

```
sin ( x )
```

Синтаксические ошибки

В компиляруемых языках синтаксические ошибки выявлятся во время компиляции программы. Компилятор может обнаружить *все* синтаксичекий ошибки в программе.

В интерпретируемых языках - во время её выполнения. При обнаружении синтаксической ошибки программа аварийно завершает свою работу. Интерпретатор как правило может указать только на одну (первую) синтаксическую ошибку.

Если же какая-то часть программы не воплнилась, то она не была проверена на наличие синтаксических ошибок.

Синтаксические ошибки. Примеры

```
sin x
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

пропущен операнд после оператора сложения

```
x + 2 = 7
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

пропущен операнд после оператора сложения

```
x + 2 = 7
```

слева от оператора присваивания обязательно должна быть только переменная!

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
    20 / 7 +
             ^
SyntaxError: invalid syntax
```

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
    20 / 7 +
             ^
SyntaxError: invalid syntax
```

Синтаксичекая ошибка в файле *program.py* в первой строчке.

Особенности синтаксиса языка Python

- После инструкций точка с запятой не ставится
- Одна строка - одна инструкция
- Конец строки - конец инструкции
- Пустые строки интерпретатором игнорируются

Особенности синтаксиса языка Python

Пример последовательности инструкций (операторов):

```
a = 42
b = 13
c = a + b

print("a + b = ", c)
```

Отдельные логические части последовательностей операторов можно разделять пустыми строками.

Для улучшения читаемости знаки операций можно отделять пробелами.

Выражения

Выражение (expression) - комбинация значений, констант, переменных, операций и функций, которая может быть интерпретирована в соответствии с правилами конкретного языка.

Арифметическое выражение - выражение результат которого имеет числовой тип (целый или вещественный)

Выражения

Примеры выражений

```
2
```

```
3 + 1
```

```
x  # x - объявленная ранее переменная
```

```
22 + x / 2 - y**0.5
```

```
a and b  # a и b переменные логического типа
```

```
sin(pi)
```

```
input()
```

Логические выражения

Логическое выражение — выражени результатом вычисления которой является «истина» или «ложь»

```
True
False
2 == 2
2 * x == 5
"war" == "peace"
not a
b or a and c
x > 10
z <= 0.5
sin(pi) > 0
```

Логические выражения

Операции с логическим типом
and, or, not, $\wedge(xor)$

```
x = 5
y = 6
z = 5

x < y and x == z   # True
x != y   # True
x == 6 or x == 7   # True
x < y < z   # False

True ^ False   # True
```

Пустой оператор

pass - оператор который не выполняет никаких действий

Используется там, где синтаксис требует наличия оператора, но алгоритм не требует выполнения действий

Outline

Оператор присваивания

Оператор присваивания (assignment statement)- оператор связывающий переменную некоторым значением.

Формат:

```
переменная = выражение
```

Во время выполнении *операции присваивания* сначала вычисляется результат выражения, а затем он связывается с переменной.

Оператор присваивания. Примеры

ПРАВИЛЬНО

```
x = 20
y = 20/8 - 6 + x

z = x

x = x + 1

y = sin( 0 )

# ничего не изменит
x = x
```

НЕПРАВИЛЬНО

```
20 = 7

x + 1 = 8

x + y = 9

sin(x) = 0
```

Особенности синтаксиса языка Python

- Отступы - это часть синтаксиса языка
- Операторных скобок нет
- Уровни вложенности определяются отступами.
- Для одного уровня вложенности рекомендуется использовать отступ из 4-х пробелов.

```
if a > b:
    print("a > b")

while a > b:
    a = a + 1
```

Перед вложенными операторами всегда ставится двоеточие.

Outline

Оператор вызова функции ()

Функция — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Формат

`идентификатор` (`параметры`)

идентификатор - идинтификатор функции;
параметры - выражения разделённые запятой.

Параметры могут отсутствовать.

Оператор вызова функции ()

Примеры

```
input()   # нет параметров

print("Python!")   # один параметр

log(9, 3)   # два параметра
# = 2.0
```

Оператор вызова функции ()

- Каждая функция имет своё количество параметров (аргументов).
- Некоторые функции могут принимать разное количество аргументов
  например print()
- Функция ождидает, что переданные ей параметры будут иметь конкретный тип
  Наприме параметров функции sin() должно быть число, а не строка
- Некоторые функции не имеют жестких требований к типы передаваемых им параметров

Outline

Условный оператор

Сокращённая форма

```
if лог.выражение :
    оператор1
```

Условный оператор

Полная форма

```
if лог.выражение :
    оператор1
else:
    оператор2
```

Условный оператор

Операторов может быть несколько.
Все они должны выделятся отступами

```
if лог.выражение:
    оператор1
    оператор2
    оператор3

оператор4
```

оператор4 будет выполнен в любом случае.

Условный оператор

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

Условный оператор

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

```
max = a
if b > max:
    max = b

print( "max = ", max )
```

Outline

Циклы

- Цикл с предусловием. while
- Цикл со счётчиком (совместный цикл). for

Циклы

Цикл с предусловием. Короткая форма

```
while условие:
    оператор1
```

Оператор1 будет выполнятся до тех пор пока условие истинно.

Циклы

Цикл с предусловием. Полная форма

```
while условие :
    оператор1
else:
    оператор2
```

Оператор2 выполнится когда условие станет ложным.

Циклы

Цикл for
Этот цикл может выполнять обход элементов последовательностей.
Например перебрать все элементы в массиве или символы в строке.

```
for e in последовательность :
    оператор1
```

Оператор1 будет выполнятся пока не будут перебраны все элементы в последовательности.
Переменную цикла e объявлять заранее не нужно.

Циклы

Создание простых последовательностей

range(n) - создаёт последовательность из n целых чисел 0..n-1
range(a, b) - создаёт последовательность из b-a целых чисел от

a..b-1
Последние число никогда не включается в последовательность.
b должно быть больше чем **a**

```
range(5)      # 0, 1, 2, 3, 4
range(0, 5)   # 0, 1, 2, 3, 4

range(3,7)    # 3, 4, 5, 6
range(-3, 1) # -3, -2, -1, 0
```

Циклы

Напечатать цифры от 0 до 10

```
for i in range(11):
    print(i)
```

Списки

В Python нет встроенного типа *массив*.
Вместо них - списки.
Синтаксис обращения к элементам списка похож на синтаксис массивов.
Объявить пустой массив:

```
l = []
```

Можно сразу задать значения:

```
l = [ 1, 2, 3 ]
```

Списку не нужно задавать начальный размер потому, что в любой момент можно добавить ещё один элемент в конец.

```
l = l + [42] ## l = [1, 2, 3, 42]
```

Списки

Некоторые операции со списками

```
l = [10, 20, 30, 42]

n = len(l)  # получить длинну списка
x = l[2]  # доступ к элементам. индексация с 0
# x = 30

z = l[-1]  # доступ к последнему элементу

l2 = [0] * 128  # создание списка из 128 нулей
```

Списки

Создать список из n случайных чисел

```
from random import random
# функция random возвращает псевдослучайное число
# в интервале от 0 до 1
```

Традиционный способ

```
n = 100
l = []
for i in range(n):
    l = l + [random()]
```

Списки

Быстрее и короче. С помощью генератора списка.

```
n = 100
l = [random() for i in range(n)]
```

Списки

Проход по списку.
"Традиционный способ"

```
l = [1,2,3,4]

for i in len(l):
    print(l[i])
```

С помошью совместного цикла

```
l = [1,2,3,4]

for e in l:
    print(e)  # в теле такого цикла нельзя менять e
```

Двумерный список

```
m = [ [1,2,3], [4,5,6], [7,8,9] ]
for l in m:
    for e in l:
        print(e, end="")
    print()
```

Ссылки и литература

ЭБС

- biblio-online.ru - ЭБС Юрайт
- studentlibrary.ru - ЭБС "КОНСУЛЬТАНТ СТУДЕНТА"

- Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата Содержит краткое описание языка.
- ru.wikibooks.org/wiki/Python - Викиучебник
- Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- Официальная документация Python3

  `help( имя )`

Дополнительная литература

- O'Conor T.J. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers. 2012 — 288 p.
- Numerical methods in engineering with Python 3 / Jaan Kiusalaas.